## <u>REMARKS</u>

Claims 4, 16, and 25 are amended to provide proper antecedent bases, mooting the § 112 rejections.

Independent claims 1, 14, 19, and 25 are amended to recite that different applications running in at least two windows in an affinity group are <u>independent</u>. Support for this amendment is found in Figure 2 and ¶ 0014, depicting a web browser, an email client, and a word processor – self-evidently independent applications – running in windows 36, 38, and 40, respectively. No new matter is added.

The Examiner rejected claims 1, 14, 19, and 25 under 35 U.S.C. § 103 as being unpatentable over U.S. Patent No. 5,995,103 to Ashe ("Ashe") in combination with U.S. Patent Number 5,920,313 to Diedrichsen *et al.* ("Diedrichsen"). To establish a *prima facie* case of obviousness, the prior art reference (or references when combined) must teach or suggest all the claim limitations. MPEP § 2143. The combination of Ashe and Diedrichsen fails to teach or suggest all limitations of claims 1, 14, 19, and 25, as amended herein.

As discussed in a prior Response, Ashe discloses a window grouping mechanism for manipulating and displaying groups of windows – all of the windows in each group associated with the same application program – via a series of linked data structures. Ashe fails to teach or suggest an affinity group of GUI windows associated with at least two different applications.

Diedrichsen discloses grouping together various child windows – those spawned by an application running in a parent window – together with the parent to form a logical group. col. 5, line 61 – col. 6, line 4. Windows in the group are identified by, *e.g.*, highlighting the parent window in high intensity and the child windows with a reduced intensity. col. 6, lines 17-34. "Thus, in a system according to the present invention, the user can always tell which objects <u>are related</u> to the selected window, even if there are more instances of the same application running." col. 6, lines 40-44.

Diedrichsen discloses only grouping windows running applications that are related as parent/child. This is clear from the mechanism Diedrichsen discloses to implement its grouping and visual marking. See, *e.g.*, Figs. 7A and 7B. Fig. 7A depicts the overall process: select an object (710); highlight it (715); and call related objects (720). Fig. 7B depicts the details of step 720. If the selected object is a parent and there are one or more child objects associated with it (740), iterate through all child objects (745, 750). On the other hand, if the selected object is a child and there is a parent associated with it (755), access its parent (760) and iterate through the parent's other child objects (765), to highlight (or otherwise mark) the group. Diedrichsen is able to iterate through these parent/child associations by pointers (created when child objects are spawned) that associate them. See Fig. 6, and col. 8, lines 22-33.

> [T]he parent window always knows about any child window it creates, and hence it can call methods on those windows to visually mark them on the display, in order to differentiate the groups of related user interface objects on the desktop; particularly, the parent window can call methods on its child windows to change the color of the window as required.

col. 8, lines 34-40. Diedrichsen discloses no other mechanism for grouping windows. In particular, Diedrichsen discloses no mechanism by which different, independent applications running in different windows may be associated to form affinity groups of windows that may be manipulated together, such as altering the z-order of the entire group on the GUI display.

Claims 1 recites "establishing, by a user, a first affinity group . . . including windows associated with at least two different, independent applications, such that the windows comprising said first affinity group are related." This defines over the combined teaching of Ashe and Diedrichsen in at least three respects.

First, the affinity group is explicitly recited as being created by a user. Diedrichsen automatically groups parent and child windows.

Second, the windows are associated with different, and independent, applications. Ashe's groups are exclusively windows forming part of a single application, and Diedrichsen's groups are exclusively windows whose applications form a family, in parent/child relationship. Neither reference suggests grouping, by a user, windows running different, independent applications. This ability provides numerous advantages over either Ashe or Diedrichsen. For example, in the situation depicted in Figure 2, a user has grouped a web browser, an email client, and a word processor, and can manipulate all three windows together. Another user may wish to form an affinity group comprising a word processor running in one window and a spreadsheet in another. The greater flexibility and wider applicability of the present invention as compared to the strictly limited window grouping functionality of either Ashe or Diedrichsen is readily apparent.

Finally, claim 1 explicitly recites that the windows comprising said first affinity group – as opposed to the applications running in them – are related. This distinction is critical, and is the key to the flexibility described above. Embodiments of the present invention allow users to group windows in a GUI environment and manipulate the windows as a group, independently of the applications running in the windows. Ashe's window grouping is limited to windows of a single application; Diedrichsen's window grouping is limited to parent/child families of applications. The claimed invention manipulates windows, and is application-independent. As such, it subsumes the functionality of both Ashe and Diedrichsen – both multiple windows of a single application and windows of applications related as parent/child may be grouped and manipulated together according to the claimed invention – but it is additionally much broader in providing the group manipulation of GUI windows independently of the applications running in them.

While the § 103 rejections have been discussed herein with respect to claim 1, Applicants note that the limitation of different, independent applications associated with windows forming an affinity group is recited in all independent claims. Neither Ashe nor Diedrichsen, separately or in combination, teach or suggest this limitation. For at least this reason, the § 103 rejections are improper and must be withdrawn. All dependent claims include all limitations of their respective parent claim(s), and thus also define patentable nonobviousness over the art of record.

All pending claims are now in condition for allowance, which prompt action is hereby respectfully requested.

Respectfully submitted,

COATS & BENNETT, P.L.L.C.

Edward H. Green

Dated: December 21, 2007

Edward H. Green, III
Registration No.: 42,604

1400 Crescent Green, Suite 300
Cary, NC 27518

Telephone: (919) 854-1844
Facsimile: (919) 854-2084